

Automatic Index Optimization combined with Text Categorization by KNN considering Feature Similarities

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we propose the KNN version which considers the feature similarities as the approaches to the text classification and the index optimization. In the previous work, we proposed the KNN version as an approach to the index optimization, but the domain should be presented as a tag, together with an input text. In the proposed system, an only text without its domain tag is given as the input, its domain is automatically assigned through the text classification, and the words from the text are classified by the classifier which corresponds to the domain. The KNN algorithm which considers the feature similarities is adopted as the approach to both tasks. The goal of this research is to automate the index optimization without presenting the domain tag and improve performance of both tasks, using the modernized KNN version.

I. INTRODUCTION

The index optimization is the process of optimizing text indexes for maximizing the information retrieval performance. Words as text indexes are defined as the three classes: expansion, inclusion, and removal; the task is interpreted into a classification task. The treatment of words each of which is classified into one of the three categories is to add associative words from external sources to ones which are classified in expansion, to take words which are classified into inclusion, and exclude ones which are classified into removal from the indexes. The classification task which is mapped from the index optimization belongs to the domain dependent task where a same entity may be classified differently, depending on the domain. It is required to present the domain as a tag for executing the index optimization.

This research is motivated by the requirement of knowing the domain in advance for doing the index optimization. The task is mapped into an independent classification task, domain by domain. It is required to present the domain as well as the text for nominating the corresponding classifier. It is discovered to automate the process of assigning a domain to a text through the text classification. To solve the problem, we connect the index optimization which is covered in this research with the text classification.

The KNN algorithm which considers the feature similarities is adopted for implementing the index optimization system which relates to the text classification. The similarity

metric between two numerical vectors which considers the feature similarities and the feature value similarities, is defined and the KNN algorithm is modified into such version based on the similarity metric. The modified KNN algorithm is applied to the text classification which automates the process of deciding the domain. A text is indexed into words, and the modified KNN algorithm is applied to the classification of each word into expansion, inclusion, or removal. This research is intended to automate the process of labeling a text with its own domain by the text classification for extracting keywords from a text.

Let us mention some benefits from this research. The similarity metric between two numerical vectors which consider the feature similarities is defined as the solution to the poor discrimination among sparse numerical vectors which represent words or texts. It is expected to improve the performance in both tasks, the index optimization and the text classification, by adopting the KNN algorithm which considers the feature similarities. It does not require to present the domain as a tag for performing the index optimization. In this research, we upgrade the index optimization system by automating the process of nominating a corresponding classifier.

Let us mention some remaining tasks as further research. We may involve only some features, rather than entire features, in computing the feature similarities, to reduce the computation time. Words and texts are encoded into compound representations, each of which consists of more than structured form. Other machine learning algorithms and deep learning algorithms may be modified into proposed versions. The index optimization system may be implemented as an independent program or a library module by adopting the proposed KNN algorithm.

II. PROPOSED APPROACH

A. Similarity Metric

This section is concerned with the process of encoding a word into a numerical vector. Texts are used as features for encoding a word so, and some are selected from a text collection. A similarity between texts is computed based on their shared words, and it is used as a similarity between

features. The similarities among features are considered for computing the semantic similarity between words. In this section, we describe the process of encoding a word into a numerical vector and the process of computing the similarity between words.

The process of encoding words into numerical vectors is illustrated in Figure 1. Texts are gathered as a collection from external sources as feature candidates. Only some texts are selected by a criterion among the gathered texts. In each word, its weights in the texts are computed as elements of the numerical vector. Refer to [2] for studying the process and the selection criteria.

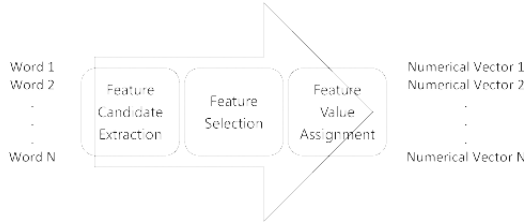


Figure 1. Process of Encoding Words into Numerical Vectors

The frame of computing the similarity between two numerical vectors is illustrated in Figure 2. The two d dimensional vectors and the d features are respectively notated respectively by $\mathbf{x} = [x_1 \ x_2 \ \dots \ x_d]$, $\mathbf{y} = [y_1 \ y_2 \ \dots \ y_d]$, and $f_1, f_2, \dots, f_d$. The feature similarity refers to the similarity between two features, which is notated by $\text{sim}(f_1, f_2)$. The feature value similarity is the similarity between two vectors, $\mathbf{x}$ and $\mathbf{y}$, such as cosine similarity. The similarity between words is computed by considering the two kinds of similarities.

$$\begin{array}{l} \mathbf{x} = [x_1, x_2, \dots, x_d] \\ \mathbf{y} = [y_1, y_2, \dots, y_d] \end{array} \quad \begin{array}{l} \text{Feature} \\ \text{Similarity} \\ \\ \text{Feature} \\ \text{Value} \\ \text{Similarity} \end{array}$$

Figure 2. Frame of Computing Similarity between Numerical Vectors

Let us mention the process of computing the similarity between numerical vectors as the semantic similarity between words. The similarity between features, f_i and f_j , $\text{sim}(f_i, f_j)$, is abbreviated into s_{ij} . The similarity between features is computed by equation (1),

$$s_{ij} = \frac{2 \times |D_i \cap D_j|}{|D_i| + |D_j|} \quad (1)$$

where D_i is the set of words in the feature, f_i , and D_j is the set of words in the feature, f_j , and the similarity matrix with the d features is constructed as a $d \times d$ square matrix,

as shown in equation (2),

$$\mathbf{S} = \begin{pmatrix} s_{11} & s_{12} & \dots & s_{1d} \\ s_{21} & s_{22} & \dots & s_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ s_{d1} & s_{d2} & \dots & s_{dd} \end{pmatrix}. \quad (2)$$

The similarity between two numerical vectors, $\mathbf{x}$ and $\mathbf{y}$, is computed by equation (3),

$$\text{sim}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^d \sum_{j=1}^d s_{ij} x_i y_j}{d \cdot \|\mathbf{x}\| \cdot \|\mathbf{y}\|}. \quad (3)$$

Equation (3) is used for computing the similarity between a novice word and a sample word in the proposed version of the KNN algorithm.

Let us make some remarks on the encoding process the process of encoding words into numerical vectors and computing the similarity between them. A word is encoded into a numerical vector by the feature candidate extraction, the feature selection, and the feature value assignment. The similarity between features which are given as texts is computed by equation (1). The similarity between numerical vectors which represent words is computed, considering the feature similarities by equation (3). In future, we will consider computing the similarities among some features rather than all features, for improving the computation speed.

B. Feature Similarity based KNN Algorithm

This section is concerned with the version of KNN algorithm which considers the feature similarities. The version was initially proposed as the approach to the text classification by Jo in 2019 ???. The similarity metric between numerical vectors which was described in the previous section is used for computing the similarity between a novice vector and a training example. The labels of its nearest neighbors are voted with their identical weights for decoding the label of a novice vector. This section is intended to describe the initial version of the KNN algorithm from which its variants are derived.

The process of computing the similarity between a novice item and a training example is illustrated in Figure 3. The labeled words which are gathered as samples are encoded into numerical vectors, and they are notated by a set $Tr = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$. A novice word is encoded into a numerical vector notated by $\mathbf{x}$. Its similarities with the numerical vectors which represent the sample words are computed by equation (3), as $\text{sim}(\mathbf{x}, \mathbf{x}_1), \text{sim}(\mathbf{x}, \mathbf{x}_2), \dots, \text{sim}(\mathbf{x}, \mathbf{x}_N)$. Some training examples which are most similar as the novice input are selected as its nearest neighbors.

In Figure 4, the process of selecting nearest neighbors by the ranking selection is illustrated. The training examples are sorted by the descending order of their similarities, after computing their similarities with a novice example. The training examples with their higher similarities with a novice

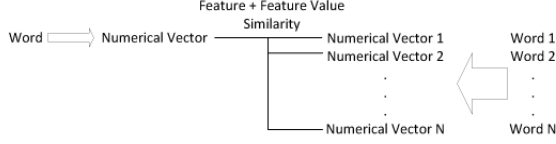


Figure 3. Similarities of Novice Input with Training Examples

example are selected as its nearest neighbors, as expressed in equation (4),

$$Nr = \text{Select}_k(Tr, \mathbf{x}), Nr \subseteq Tr \quad (4)$$

The set of nearest neighbors, Nr , is composed with elements as expressed in equation (5),

$$Nr = \{\mathbf{x}_1^{near}, \mathbf{x}_2^{near}, \dots, \mathbf{x}_k^{near}\} \quad (5)$$

The elements in the set, Nr , are used for voting their labels in the next step.

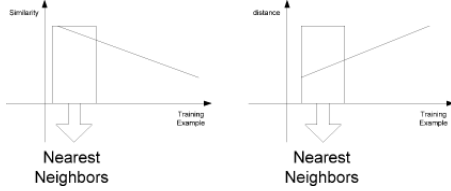


Figure 4. Selection of Most Similar or Least Distant Training Examples as Nearest Neighbors

The process of voting the labels of nearest neighbors for decoding the label of a novice word is illustrated in Figure 5. The predefined categories are notated by $c_1, c_2, \dots, c_m$, and the label of a nearest neighbor is notated by $c_j = \text{label}(\mathbf{x}_i^{near})$. The number of nearest neighbors which belong to the category, c_j , is notated by $\text{Count}(Nr, c_j)$. The label of the novice item, $\mathbf{x}$, is decided by equation (6),

$$\text{label}(\mathbf{x}) = \arg\max_{j=1}^m \text{Count}(Nr, c_j) \quad (6)$$

The process of deciding the label of a novice item by equation (6), is called voting.

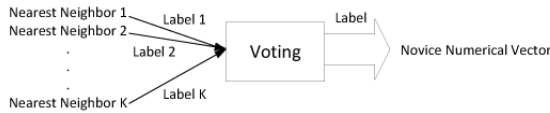


Figure 5. Voting of Labels of Nearest Neighbors for Deciding Label of Novice Input

Let us make some remarks on the proposed version of KNN algorithm which considers the feature similarities. It computes the similarities of a novice item with the training examples. The training examples with their highest similarities with the novice item are selected as its nearest neighbors. The label of the novice item is decided by voting

the labels of its nearest neighbors. The ranking selection is adopted for selecting the nearest neighbors from training examples, in this version.

C. System Architecture

This section is concerned with the index optimization system which adopts the KNN algorithm which considers the feature similarities. In Section II-B, we described the proposed version of KNN algorithm as the approach to the index optimization. It is viewed as a classification task which classifies a word into one of the three categories. This section is intended to describe the index optimization system with respect to its function and its architecture.

The process of collecting the same words and encoding them into numerical vectors for implementing the index optimization system is illustrated in Figure 6. The index optimization is interpreted into a domain dependent classification; even a same word is classified differently, depending on the domain. For each domain, an independent classification task is defined. The several domains are decided, and the words which are labeled with one of the three categories exclusively in each domain. It is required to present the domain from a text which is given as the input for doing this task.

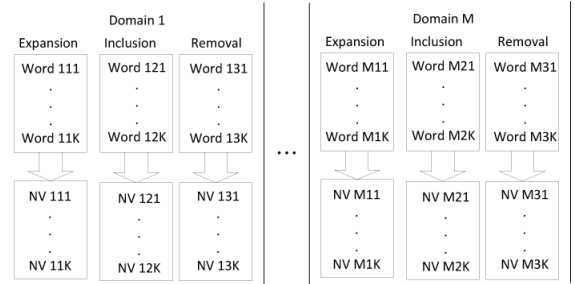


Figure 6. Preparation of Training Examples

The entire system architecture of the index optimization system which is implemented in this research is illustrated in Figure 7. The words which are labeled one of the three categories are collected as the sample words, and they are encoded into numerical vectors by the encoder module. A $d \times d$ similarity matrix which is used for computing the feature similarity is constructed, and the similarities of each word with the sample words which is indexed from a text are computed in the similarity computation module. In the voting module, nearest neighbors are selected, and the category is decided for each word in the text. In the similarity computation module, the ranking module is nested for selecting the nearest neighbors.

The execution process of the proposed system is illustrated as a block diagram in Figure 8. Sample words which are labeled with one of the three categories are collected within a domain, and they are encoded into numerical vectors. An input text is indexed into a list of words, and

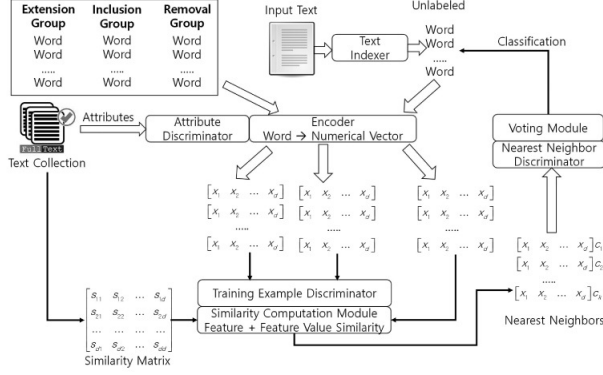


Figure 7. System Architecture

they are also encoded into numerical vectors. The nearest neighbors are selected by the similarity computation, and the label is decided by voting ones of their nearest neighbors for each word. Ones which are classified into expansion require their associated words from external sources, ones which are classified into inclusion are preserved, and ones which are classified into removal are excluded.

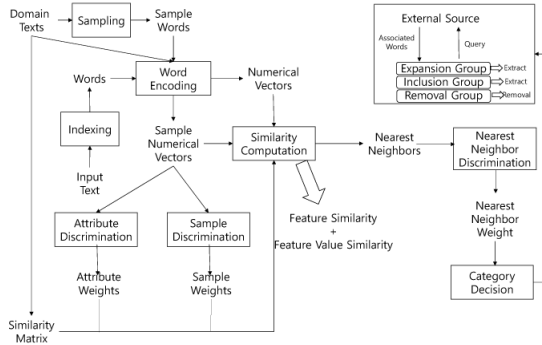


Figure 8. System Flow

Let us some remarks on the index optimization system which is presented in Figure 7 and 8. We need the two collections: the collection of sample words and the collection of texts within each domain. The system is implemented as multiple independent subsystems domain by domain, and it is required to present the domain as a tag for performing the index optimization to a text. The proposed system is described staying in the design step; the source code in Java or Python will be presented in the next research. We need to implement the additional module for expanding the index to the words in the expansion group.

D. Connection with Text Classification

This section is concerned with the connection of the index optimization with the text classification. In the previous section, we mentioned the index optimization system as an instance of domain dependent classification. The domain is

automatically decided for nominating a classifier by connecting the keyword with the text classification. The KNN algorithm which was described in Section II-B is used for implementing the text classification. This section is intended to describe the connection of the index optimization system with the text classification for automating the decision of the domain.

The system architecture of the text categorization system which consists of the encoding module, the similarity computation module, and the voting module is illustrated in Figure 9. The encoding module is for encoding texts into numerical vectors by the process which is described in Section II-A. The similarity computation module is for computing the similarities between numerical vectors which represent a novice text and a sample text and selecting ones with their highest similarities as the nearest neighbors. The voting module is for deciding the label of the novice item by voting the labels of the nearest neighbors. This system used for automating deciding the domain of the input text.

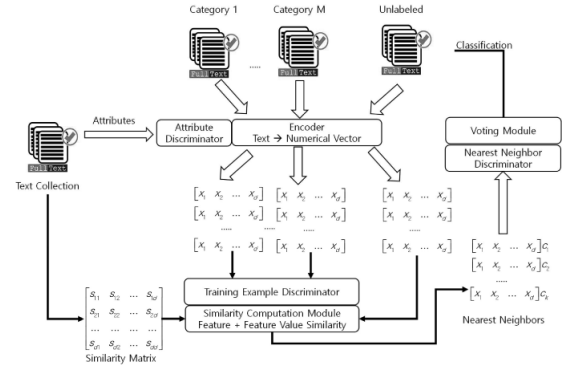


Figure 9. Text Categorization System

The connection of the index optimization with the text classification is illustrated in Figure 10. The texts each of which is labeled with its own domain are prepared as the sample texts, and the approach to the text classification is trained with them. A domain is assigned to a novice text by classifying it into a domain automatically. The novice text is provided with its domain to the index optimization system which is described in Section II-C. The role of the text classification system is to nominate the classifier which corresponds to the domain for the index optimization, automatically.

The process of executing the index optimization, connecting it with the text classification is illustrated in Figure 11. The sample texts each of which is labeled with its own domain are prepared, and the sample words each of which is labeled with one of the three categories are prepared in each domain. A novice text is classified into one among the predefined domains, and the classifier which corresponds to the domain. The novice text is indexed into a list of words, and each word is classified into one of the three categories.

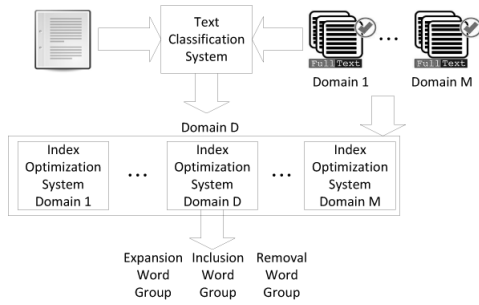


Figure 10. Index Optimization System connected with Text categorization

The labeled words are the final output from this system; the domain which is assigned to the input text is the guide for selecting a classifier.

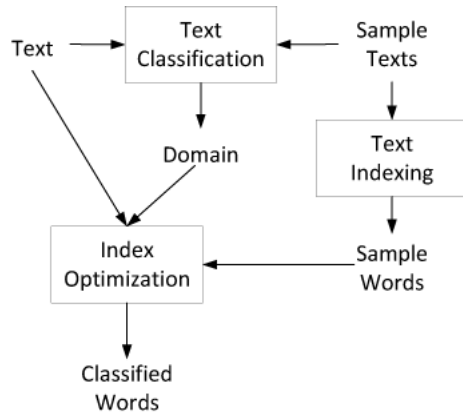


Figure 11. System Flow of Index Optimization connected with Text Categorization

Let us make some remarks on the connection of the index optimization with the text classification. The text classification is applied to the automatic decision of the input text domain. The role of text classification is to decide the domain for nominating a classifier, and the role of the index optimization is to classify words into expansion, inclusion, or removal. In the execution flow, the input text is classified into a domain, it is indexed into a list of words, and each word is classified into one of the three categories. We consider connecting the text classification as the main task the index optimization as the opposite to what is proposed in this research.

REFERENCES

- [1] T. Jo, "Index Optimization in News Articles using Feature Similarity based K Nearest Neighbor", 106-109, The Proceedings of 17th Int'l Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government, 2018.
- [2] T. Jo, "Text Classification using Feature Similarity based K Nearest Neighbor", 13-21, AS Medical Science, 3, 4, 2019.